Climate Ocean Modeling on Parallel Computers

Ping Wang
Jet Propulsion Laboratory
California Institute of Technology
MS 168-522, 4800 Oak Grove Drive
Pasadena, CA 91109-8099, U.S.A.
wangp@rockymt.jpl.nasa.gov
http://www-hpc.jpl.nasa.gov/PEP/wangp

Benny N. Cheng
Jet Propulsion Laboratory
California Institute of Technology
MS 300-323, 4800 Oak Grove Drive
Pasadena, CA 91109-8099, U.S.A.
bnc@pacific.jpl.nasa.gov
http://oceans-www.jpl.nasa.gov/hpcc/cheng.html

Yi Chao
Jet Propulsion Laboratory
California Institute of Technology
MS 300-323, 4800 Oak Grove Drive
Pasadena, CA 91109-8099, U.S.A.
yc@comp.jpl.nasa.gov
http://comp.jpl.nasa.gov/~yc/YC/

parallel computers

Abstract:

Ocean modeling plays an important role in both understanding the current climatic conditions and predicting future climate change. However, modeling the ocean circulation at various spatial and temporal scales is a very challenging computational task. In contrast to the atmosphere, where the dominant weather system has a spatial scale of 1000s km, much of the ocean energy is associated with mesoscale eddies (equivalent to the storms in the atmosphere) with a spatial scale of 100s km near the equator to 10s km at high latitudes. With an order of magnitude smaller in both longitudinal and latitudinal directions and a few time shorter in the temporal scale, a minimum ocean model is at least 100 times more computationally demanding than a typical atmospheric model. Implementing a well-designed parallel ocean code and improving the computational efficiency of the ocean model will significantly reduce the total research time to complete these studies. There are many challenges to design an efficient ocean modeling code on parallel systems, such as how to deal with an irregular computing geometry on a parallel system and how to port a code from one system to other systems.

This chapter reports on our efforts to implement an efficient parallel ocean model on distributed memory and shared memory systems.

The computation domains of an ocean model is usually irregular. A simple method to deal with irregular geometries is to use a rectangular geometry to approach the irregular domain. In this case, many processors will be idle on land while other processors are working on the ocean domain. For some applications, it will cause more than 30% processors to be idle. We have developed a flexible partitioning technique for irregular ocean geometries on parallel systems. It has 2D partitioning features and works on any irregular geometry, and the communication pattern on the subdomains has been designed as a virtual torus. MPI software is used for communication which is required when subdomains on each processor need neighboring boundary data information. This partitioning structure has dramatically saved computing resources, leading to a significant speed up. Because of the portability of this software, the code can be executed on any parallel system which supports MPI library.

We want to emphasize the portability of the ocean model across a variety of parallel platforms, ranging from the most powerful supercomputers to the affordable desktop parallel PC cluster (Beowulf-class system). To this end, we have successfully ported the most widely used three-dimensional time-dependent ocean general circulation models to various parallel systems, including the scalable parallel systems Cray T3E-600, the shared memory system HP Exemplar SPP-2000, and the 16-node PC cluster Beowulf system. Procedures on how to implement and optimize the code to different system are discussed, and intensive comparisons of wallclock time with various grid sizes are made among several parallel systems. Scientific results from a Atlantic ocean model with high resolutions have been obtained using 256 processors on the Cray T3D.

Keywords:

ocean modeling, parallel computation, partition, parallel systems, irregular geometries

1. INTRODUCTION

Ocean modeling plays an important role in both understanding the current climatic conditions and predicting the future climate change. However, modeling the ocean circulation at various spatial and temporal scales is a very challenging computational task. In contrast to the atmosphere, where the dominant weather system has a spatial scale of 1000s km and a temporal scale of weeks, much of the ocean energy is associated with mesoscale eddies (equivalent to the storms in the atmosphere) with a spatial scale of 100s km near the equator to 10s km at high latitudes and a temporal scale of days. With an order of magnitude smaller in both longitudinal and latitudinal directions, a minimum

ocean model is at least 100 times more computationally demanding than a typical atmospheric model.

Thus, it was not until recently that eddy-permitting (or eddy-resolving) calculations could be carried out on a basin or global scale. Using the vector supercomputers (e.g., Cray Y-MP) at National Center for Atmospheric Research (NCAR), decade-long ocean model integrations have been carried out at 1/4 degree horizontal resolution [1], which was the first ocean model with performance exceeding 1 billion floating-point-operation-per-second (1 Gflops/s). With the advance of massively parallel computing technology, decade-long integrations at 1/6 degree resolution have been conducted at Los Alamos National Laboratory (LANL) [2] and Jet Propulsion Laboratory (JPL) [3] on the CM-5 and Cray T3D, respectively. Recently, a 10-year integration at 1/10 degree resolution was made at LANL using CM-5, and a six-year integration at 1/12 degree resolution was made on the T3D at the Pittsburgh Supercomputer Center [4]. Despite the recent progress in eddy-resolving ocean modeling, it is apparent that we are far from convergence in resolution, because each of these higher resolution calculations show additional features that were not resolved in a coarser resolution model. Furthermore, the increase of spatial resolution sometimes can even degrade the solution at coarser resolutions, suggesting the need of more experimentation of eddy-resolving models.

In this paper, we report on our experiences running one of the most widely used ocean models on a variety of parallel computer systems. One of our objectives is to improve the computational efficiency of the ocean model on parallel computers such that one can reduce the time in conducting scientific studies. We also want to emphasize the portability of the ocean model. Experiences of porting and optimizing the ocean model on a variety of parallel systems are also described. With an efficient ocean model on multiple platforms, one can maximize the limited computational resources available for climate ocean modeling.

2. MODEL DESCRIPTION

The ocean model used in this study is among one of the most widely used OGCM code in the community. The OGCM is based on the Parallel Ocean Program (POP) developed at Los Alamos National Laboratory[2]. This ocean model evolved from the Bryan-Cox 3-dimensional primitive equations ocean model [5, 6], developed at NOAA Geophysical Fluid Dynamics Laboratory (GFDL), and later known as the Semtner and Chervin model or the Modular Ocean Model (MOM) [7]. Currently, there are hundreds of users within the so-called Bryan-Cox ocean model family, making it the most dominant OGCM code in the climate research community.

The OGCM solves the 3-dimensional primitive equations with the finite difference technique. The equations are separated into barotropic (the vertical mean) and baroclinic (departures from the vertical mean) components. The baroclinic component is 3-dimensional, and uses explicit leapfrog time stepping. It parallelizes very well on massively parallel computers. The barotropic component is 2-dimensional, and solved implicitly. It differs from the original Bryan-Cox formulation in that it removes the rigid-lid approximation and treats the sea surface height as a prognostic variable (i.e., free-surface). The free-surface model is superior to the rigid-lid model because it provides more accurate solution to the governing equations. More importantly, the free-surface model tremendously reduces the global communication otherwise required by the rigid-lid model.

Building upon the original ocean model developed at LANL, the new JPL ocean model has significantly optimized the original code [8]. In that chapter, several optimization strategies were described including memory optimization, effective use of arithmetic pipelines, and usage of optimized libraries. The optimized code runs about 2.5 times faster than the original code, which corresponds to 3.63 Gflops on the 256-PE Cray T3D. A user-friendly coupling interface with the atmospheric or biogeochemical models was also developed [9]. Such a model improvement allows one to perform ocean modeling at increasingly higher spatial resolutions for climate studies. In this chapter, we focus on how to deal with an irregular computing geometry on a parallel system and how to port a code from one system to other systems.

3. PARALLEL PARTITION ON IRREGULAR GEOMETRIES

In our present work, we chose the most widely used OGCM code as our base code. This OGCM is based on the Parallel Ocean Program (POP) developed in FORTRAN 90 on the Los Alamos CM-2 Connection Machine by the Los Alamos ocean modeling research group. During the first half of 1994, the code was ported to the Cray T3D by Cray Research using SHMEM-based message passing. Since the code on the T3D was still time-consuming when large problems were encountered, improving the code performance was considered essential. Recently much effort has been taken to optimize this code [8]. A detailed description about optimization strategies was reported in this paper. Such a model improvement allows one to perform ocean modeling at increasingly higher spatial resolutions for climate studies. But there are still many challenges for ocean modeling on parallel systems. How to deal with an irregular computing geometry on a parallel system is one of important issues we like to address in this paper.

Domain decomposition techniques are widely used in parallel computing community. This method is simplely to split an original computation domain to N subdomains such that each processor can just work on one subdomain. Once N processors are applied, a

significant speed up should be achieved-a total cpu time T is reduced to T/N. In order to design a parallel code by use of the domain decomposition techniques, partition of a computation domain is the first issue encountered. Computation domains of ocean models are usually irregular. A simple method to deal with irregular geometries is to use a rectangular geometry to approximate an irregular domain. In this case, a partitioning structure can be easily designed according to a regular node mesh which can be a 1D,2D, or 3D partition.

Here a simple 2D partition is given. For an M-by-N 2-dimensional partition, in which (M*N).EQ. NPES (total processors), neighbors are to the north, south, east, and west. The following example first converts MY-PE (current processor) to coordinates in the abstract topology(2D torus):

I=MOD(MY_PE,M)

J=MY_PE/M

The coordinate of MY_PE is the abstract topology is (I,J). I and J were chosen in such a way that MY_PE.EQ.(I+M*J).

This kind of partitions gives a 2D mesh which divides a rectangular domain to M*N subdomains such that each processor only needs to work on one subdomain. But for ocean modeling, there are nearly no rectangular geometries. If this simple 2D partition is applied, computing efficiency will deteriorate as some processors are idle on non-ocean domains which are inside the 2D mesh domain. For instance, the original POP code used a 2D partition to divide the 3D ocean domain into subdomains in a rectangular computing geometry. Figure 1 shows the 2D partition on the North Atlantic ocean. Here about 36% of the processors are idle on land while other processors are working on the ocean domain. So designing a flexible partitioning structure, which works only on ocean area, will dramatically save computing resources, leading to significant speed up via an efficient partition.

Recently we have developed a new irregular partitioning structure for ocean modeling. In order to achieve load balance, and to exploit parallelism as much as possible, a general and portable parallel structure based on domain decomposition techniques is designed for the three dimensional ocean domain. It has 2D partitioning features and works on any irregular geometry, and the communication pattern on the subdomains has been designed as a virtual torus. MPI software is used for communication which is required when subdomains on each processor need neighboring boundary data information. Because of the portability of this software, the code can be executed on any parallel system which supports MPI library.

The main idea of this approach is to eliminating idle processors from the original 2D rectangular partition and to find the nearest neighboring processors which are active. Assume NPES are used for computation by use of a rectangular 2D mesh MxN. Since we know the irregular ocean topography G, it is easy to calculate total idle processors K caused by the 2D mesh. A target of total processors should be equal to NPES-K if all processors only work on ocean domain. Now the major problem is how to find locations of the nearest neighboring non-idle processors for each active processor. A detailed procedure of locating neighboring processors is given in the following.

a . Given an original total number of processors NPES, 2D mesh MxN, ocean topography G, Calculate total idle processors K and get a new target number of total processors NEWPES.

b. Calculate idle processors mesh IDLE as the following:

IDLE(i,j)=0 if processor is active

IDLE(i,j)=1 if processor is idel

Here i,j are index for the original 2D mesh.

c. For each PE, calculate the total number of idle processors which index are smaller than MY_PE(i,j), and save as ITOTAL(i,j).

d. Calculate neighboring processor's locations for each PE by

EAST=MY_PE(i+1,j)-ITOTAL(i+1,j)

WEST=MY_PE(i-1,j)-ITOTAL(i-1,j)

NORTH=MY_PE(i,j+1)-ITOTAL(i,j+1)

SOUTH=MY_PE(i,j-1)-ITOTAL(i,j-1)

e. Modify boundary conditions and other input data files.

f. Perform ocean modeling on MPES-K processors by use of new neighboring information computed in (d) and MPI software for communication.

The above procedure is very straight forward, efficient, and easy to use. We have successfully applied this procedure to the ocean modeling code for the North Atlantic

Ocean (Figure 2), and it gives excellent results. All idel processors are eliminated, and the computing efficiency has been improved by 36%. This simple approach can be easily used to any problem which has an irregular computing domain and parallelized by domain decomposition techniques on parallel systems, and it leads significant speed up.

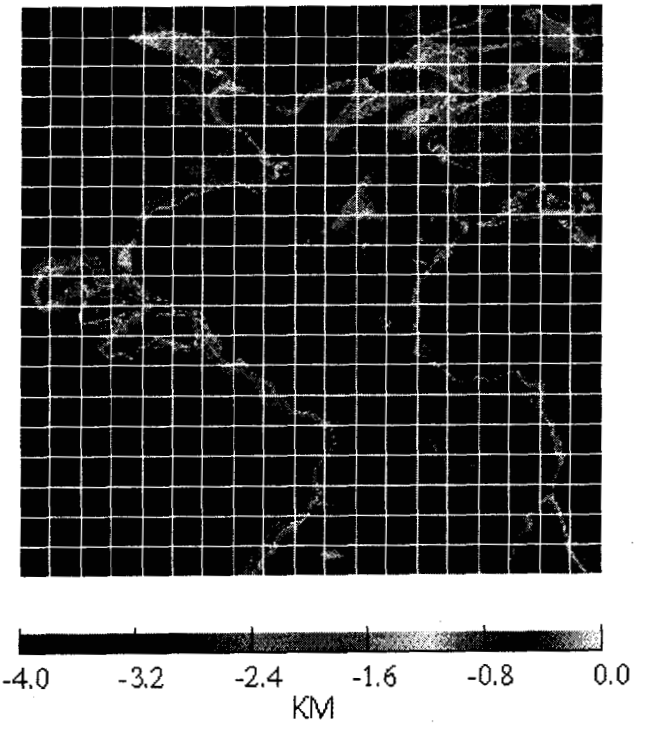

Figure 1. The original 2D partition on the North Atlantic ocean.

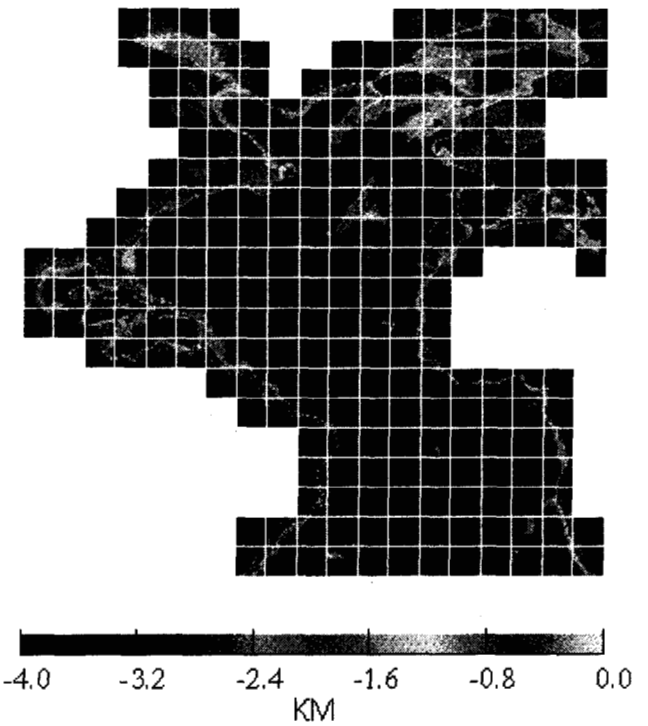

Figure 2. The new 2D partition on the North Atlantic ocean.

4. PARALLEL COMPUTATION ON VARIOUS SYSTEMS

Three-dimensional time-dependent ocean models require a large amount of memory and processing time to run realistic simulations. The significant computational resources of massively parallel supercomputers promise to make such studies feasible. But the dedicated cpu time needed by the user to run the model is quite limiting, as most systems are shared by multiple users, and access to an alternative system should ease the computational traffic jam. Therefore, we need to make the code portable to various systems in order to take advantage of various available computing systems.

4.1 Parallel systems

In order to port the ocean code to different systems, understanding current advanced parallel systems, ranging from the most powerful supercomputers to the affordable desktop parallel PC cluster (Beowulf-class system), is necessary. Here four typical parallel systems, the Cray T3D, the Cray T3E, the HP SPP2000, and the Beowulf cluster system, are considered. A brief description of those systems, which are the major computing resources used for the present study, is given here. Major features for each system are summerized in Table 1.

The Cray T3D at Jet Propulsion Laboratory (JPL), a scalable parallel system, has 256 compute nodes with 150 Mflops peak performance and 64 Megabytes memory per node. Logically, it has a shared memory, and physically a distributed memory, associated with a processor.

The Cray T3E at the Goddard Space Flight Center, currently one of the most powerful MIMD computers available, has 1024 compute nodes with 600 Mflops peak performance and 128 Megabytes memory per node. It is a scalable parallel system with a distributed-memory structure. This machine improves application performance by three to four times over that of the previous generation Cray T3D.

The 256 HP SPP2000 (Exemplar) at JPL and California Institute of Technology has 16 hypernodes, with each hypernode consisting of 16 PA-8000 processors and a single pool of 4 GB of shared physical memory. The overall architecture of the Exemplar is a hierarchical Scalable Parallel Processor(SPP). The topology of the 16 hypernodes connected by CTI(Coherent Toroidal Interconnect) is a 4x4 toroidal mesh. The Exemplar supports a variety of programming models including global shared memory programming models and explicit message passing models.

The Beowulf system at JPL, has 16 processors interconnected by 100 base T Fast Ethernet. Each processor includes a single Intel Pentium Pro 200 MHz microprocessor which has a peak speed of 200 Mflops, 128 Megabytes of DRAM, 2.5 GBytes of IDE disk, and PCI bus backplane, and an assortment of other devices. It is a loosely coupled, distributed memory system, running message-passing parallel programs that do not assume a shared memory space across processors.

All those systems support a Fortran 90 compiler and MPI software. The Beowulf system at JPL runs the Linux operating system, and NAG Fortran 90 compiler is installed in this system. Other systems run their own operating systems and Fortran 90 compilers. Since all above systems support explicit message passing models, theoretically, any Fortran 90 code designed by the domain decomposition techniques and MPI software should be able to execute on those system.

Platform	Total nodes	Mflops each node	Memory	Site
Beowulf	16	200	128 MB /node	
Cray T3D	256	150	64 MB /node	
Cray T3E-600	1024	600	128 MB /node	
HP Exemplar SPP-2000	256	720	64 GB total	

Table 1. Parallel computing systems.

Recently we have ported the North Atlantic POP code to several distributed memory and shared memory systems, including the four systems described above. Porting a code to different systems needs some basic knowledge about the hardware and its software as well. The ocean code runs well on some platforms, such as CM-2 Connection Machine, but it was not a simple issue to compile and run the ocean code on a new system. Recent developed compilers for each parallel system have much more strict rules for source codes comparing with earlier developed compilers. To run the ocean code on recently developed computing systems produces challenges. Each compiler on each system has its own features, so some modifications for makefile files and source codes are required for porting a code from one system to another. Many problems were encountered during the process of porting the code, and some major problems are reported in the following.

The compilation of ocean code on various parallel systems was the first problem encountered. Many error messages were generated at the initial compilation on a new system. Problems also vary on each systems. Details for porting the code from one system to another system are described in the following.

4.2 Ocean modeling on the Cray T3E

During the process of porting the optimized POP code from the Cray T3D [8] to the Cray T3E, Several problems were encountered at the initial compilation. First, the assembler code for the Cray T3D could not used for the Cray T3E; it had to be replaced by Fortran calls. This was easily accomplished by utilizing some of the fortran calls originally written for the CM5 machine. Most of the assembly calls originated in the stencils.f routines, and the replacement Fortran codes were subsequently optimized for the Cray T3E. After other changes of the source code, the code was compiled successfully. But the code ran slower than that on the Cray T3D, which was far behind the estimated speed up on the Cray T3E. Theoretically, the Cray T3E should improve application performance by three to four times over that of the previous generation Cray T3D because of the the hardware differences. This is due to the replacement of the assembler code with Fortran calls, and the entire code was not optimized on the Cray T3E.

Next, optimization of the code was considered for improvement of code performance. The STREAMS environment was turned on via the setenv STREAMS = 1 . This resulting in timing runs par with the optimized T3D code. At the same time, the optimization options "-scalar3 -unroll3" were used; changes were made for BLAS (fast math) routines by use of shmem calls as the BLAS library is not optimized on the Cray T3E. At this stage, the updated code ran about two times faster than the code on the Cray T3D.

Another major speedup was obtained by replacing all the default double precision reals with single precision reals. This was done using a script to automate the conversion, and resulting in another 50% increase in FLOPS performance. After all those modifications, the code ran about four times faster than the code on the Cray T3D. The final version was tested for an application on the North Atlantic region. It was running successfully; results are given in the next section.

4.3 Ocean modeling on the HP SPP2000

Porting the POP code from the standard distribution at LANL to the HP SPP2000 was also considered. The POP code with MPI version developed for the SGI machine was taken as the basic code. Since the code contains mixed single and double precision variables, it might cause some problems when MPI calls are used. Implementation of all variables as double precision (real*8) variables was necessary. A script was written to convert all single precision variables to double precision ones. Subroutine SIGNAL calls were replaced by function calls instead. Calls to SGI timer routines were replaced by corresponding calls to MPI timers. IO routines were modified to reflect the MPI topology specified in the ocean model setup. Finally, specific HP X-class exemplar compiler's optimization options " +DA2.0N +DS2.0a -O3 +Oaggressive +Odataprefetch" were applied to speed up the code. The modified code ran well on this system, and results are given in the next section.

4.4 Ocean modeling On the Beowulf system-PC clusters

The code for the HP SPP2000 was also ported to the Beowulf cluster system running on Linux. The NAG Fortran 90 compiler was used. Options " -mismatch_all -dusty" had to be applied for compilation. Minor changes for the source code were made, such as the signal calls and timing calls. Double precision was applied for all REAL variables. After those modifications, the code was compiled successfully. Then the Option "-O" was used for automatical optimization to improve the code performance, and it ran well on the Beowulf system.

4.5 Code performance and discussion

Currently the ocean code has been successfully ported to several parallel systems. No doubt, besides understanding an original code, some knowledge of new compilers and new computers will save plenty of time to port an existing large code to a new computing platform. After all those efforts, the POP code with the new partition, with a virtual torus topology described in the previous section, executes well on the HP SPP2000, the Cray T3E, T3D, and the Beowulf system, and it is easy to be ported to any parallel system which supports MPI software.

Various code performance tests have been carried out among those systems. In order to maximize the performance of the ocean code, the use of parallel software tools and compiler options have been fully explored. For example, on the HP Exemplar the options " -O3 +Oaggressive +Odataprefetch " are used for runing the ocean code. In order to compare the performance on each system, 16 processors are used for the parallel code due to the current maximum number of nodes on the JPL Beowulf system. Here a model with grid sizes 180x180x20, 360x180x20,360x360x20, and 720x360x20 is tested on those machines.

The results are shown in Figure 3, which lists the wallclock time at fixed time steps on the test problem for the four systems described earlier. The HP Exemplar gives the best performance, and the Cray T3E shows better performance than the Cray T3D and the Beowulf. But it is interesting to note that the 16-node Beowulf system is slightly faster than the Cray T3D, and the difference is growing with the grid size used. This is due to each PC of the Beowulf system is faster than the Cray T3D single node, but the communication on the Beowulf is slower than that on the Cray T3D. Overall, for a moderate grid size like 180x180x20, the communication plays a significant role. Hence, the discrepancy in time for the entire computation on the four systems depends on the difference in the network connections and the hardware. Once the grid size increases, the computation becomes dominant. Then the discrepancy in time is more consistent with the differences in the hardware.

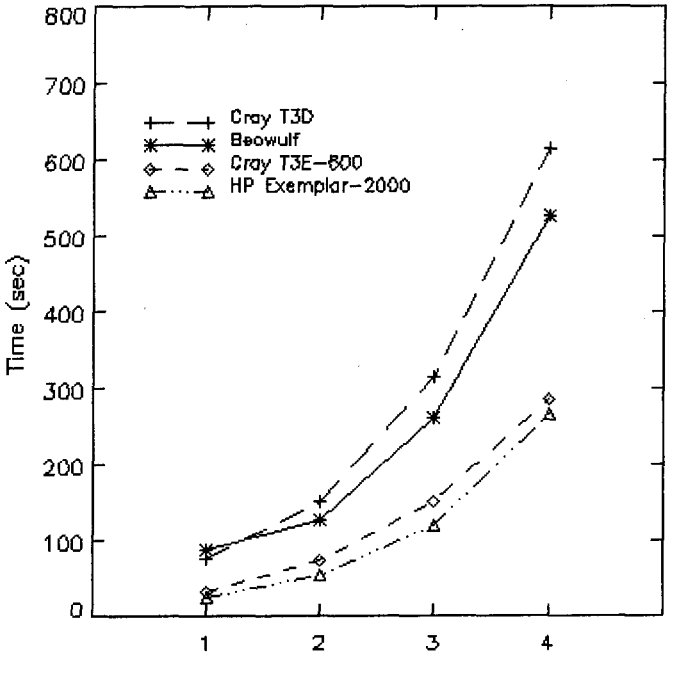

Figure 3. Ocean modeling on different parallel systems with various grid sizes (1: 180x180x20, 2: 360x180x20, 3: 360x360x20, 4: 720x360x20).

As shown in Figure 3, the results on the Beowulf system are very promising: the code runs slightly faster than that on the Cray T3D and also reaches half the speed of the Cray T3E and HP Exemplar. These results are very interesting for the scientific computing community because of the low cost of the Beowulf system. Right now, the Pentium II 300 Mhz or 400 Mhz is available, and a faster network at gigabits rate of transfer instead of 100 mbits is available at a reasonable price. More importantly, the price for these commodity products will drop dramatically with time, so it is a very cost effective way to perform large-scale scientific computing. It is quite feasible to build a dedicated Beowulf parallel system for a specific application such as ocean modeling at a fraction of the price for commercial parallel systems.

5. SCIENTIFIC RESULTS

Using 256-processor Cray T3D, we have conducted a 40-year integration of a Atlantic ocean model with a spatial resolution of 1/6 degree and 37 vertical level. Figure 4 shows a snapshot of the 3-D volume of the velocity The detailed description of the visualization method is described in [9]. From Figure 4, it is seen that the water originated in the tropics is carried northward by the western boundary current. It passes through the Caribbean Sea, the Gulf of Mexico, the Florida Straits, and subsequently the Gulf Stream. Along the way, the strong western boundary current produces mesoscale eddies. This western boundary current plays a key role in the heat transport in the North Atlantic ocean. Currently, we are coupling this 1/6 degree Atlantic Ocean model with a global atmospheric model on the Cray T3E. With the much bigger memory on the HP SPP-2000, it is anticipated that we can run a 1/12 degree Atlantic Ocean model.

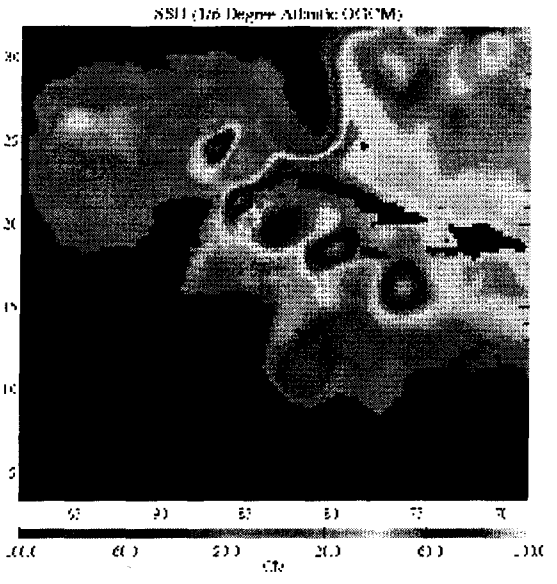

Figure 4.Snapshot of the 3-D volume of the velocity.

The velocity magnitude is linearly mapped to a spectrum color map with purple for the lowest value and red for the highest value. The opacity of the voxel is also linearly determined by the velocity. The voxels with high velocity are more opaque and the ones with low velocity are mostly transparent. The white contour map underneath the 3-D velocity is the ocean floor topography.

In a regional application, we have studied the temporal and spatial evolution of mesoscale eddies in the Gulf of Mexico and the Caribbean Sea(Figure 5). The eddies in the Caribbean Sea are quite regular, appearing about every 2~3 months. The eddies in the Gul of Mexico, on the other hand, are found about every 10 months. In comparison with observations, the above described 1/6 degree North Atlantic ocean model is able to reproduce major features of these eddies in the Gulf of Mexico and Caribbean Sea, including their amplitudes, spatial and time scales, and propagation speed. Accurate description and understanding of these eddies in the Gulf of Mexico and Caribbean Sea are crucial for coastal monitoring and forecasting, which are of great benefit to regional fishery and oil industries.

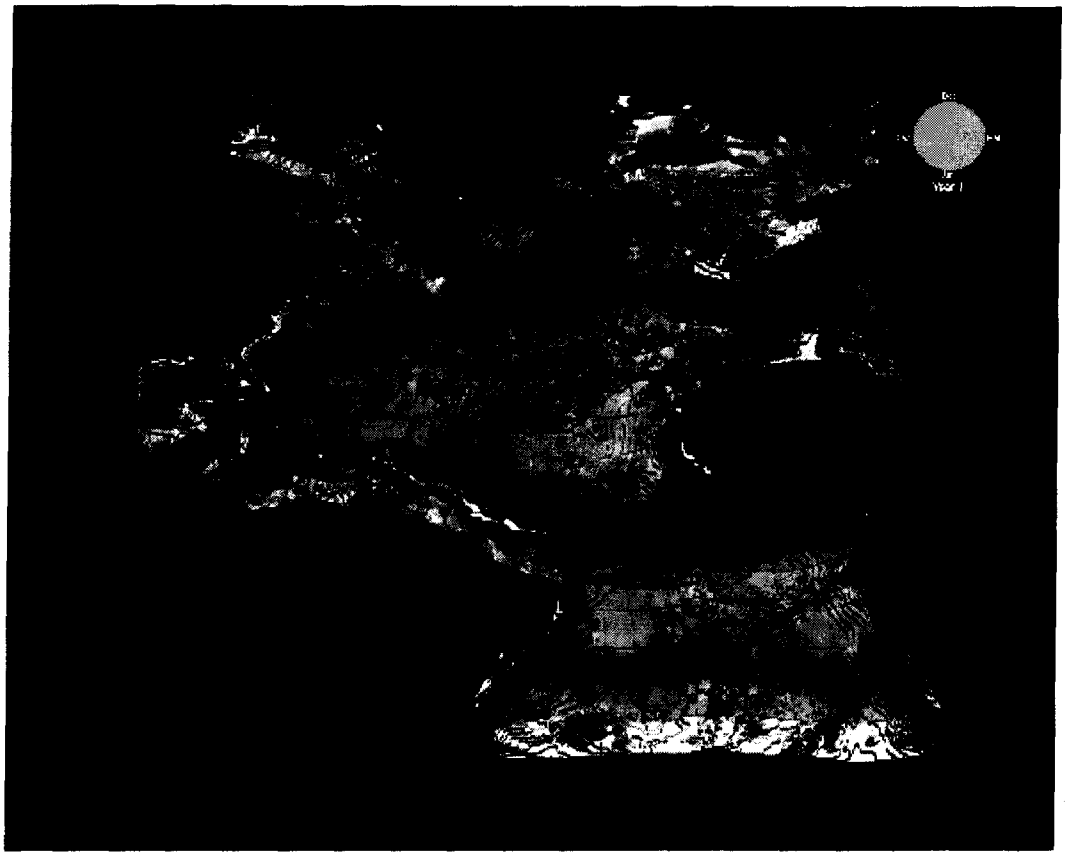

Figure 5. The sea surface height over the Gulf of Mexico and the Caribbean Sea simulated by the 1/6 degree Atlantic Ocean model.

6. CONCLUSIONS

In the present study, we have developed an efficient, flexible, and portable parallel partitioning structure which can be used for any irregular ocean geometry. This feature allows us to study various ocean problems with different geometries on parallel systems. We have also successfully ported the ocean code to various distributed memory and shared memory systems. Detailed description of porting the ocean code from one system to another system has been reported. The comparison of wallclock time for fixed time steps among these systems gives very useful information on the speedup performance of these advanced hardware systems. The discrepancy of the time on these systems is due to the difference of the hardware on each system, and the network connection used. The code scales very well on different system as the problem's size increases with a total fixed number of processors. The code can be easily ported to any parallel system which supports a Fortran 90 compiler and MPI software, in particular, the Beowulf system (pile of PCs) makes high resolution ocean computing a reality to the low cost parallel supercomputing community. Interesting scientific results have been obtained from the North Atlantic ocean model using a large number of processor. In spite of the difficulties

associated with high resolution simulation of ocean model, our present results illustrated here clearly demonstrate the great potential for applying our current approach to solving much more complicated ocean flow in realistic, time-dependent, three-dimensional geometries using parallel systems with large gird sizes.

References

1. A.J. Semtner and R.M. Chervin, **Ocean-General Circulation from a Global Eddy-Resolving Model**, *J. Geophys. Research Oceans*, 97, 5493-5550, 1992.

2. R.D. Smith, J.K. Dukowicz, and R.C. Malone, **Parallel Ocean General Circulation Modeling**, *Physica D*, 60, 38-61, 1992.

3. Y. Chao, A. Gangopadhyay, F.O. Bryan, W.R. Holland, **Modeling the Gulf Stream System: How Far From Reality?**, *Geophys. Res. Letts.*, 23, 3155-3158, 1996.

4. R.D. Smith, Los Alamos National Laboratory, and E. Chassignet, University of Miami, 1998, personal communication.

5. K. Bryan, **Numerical Method for the Study of the World Ocean Circulation**, *J. Comp. Phy.*, 4, 1687-1712, 1969.

6. M.D. Cox **Primitive Equation, 3-Dimensional Model of the Ocean**, *Group Tech. Report 1, GFDL/NOAA, Princeton, NJ*, 1984

7. R. Pacanowski, R.K. Dixon, and A. Rosati, **Modular Ocean Model User's Guide**, GFDL Ocean Group Tech. Report 2, GFDL/NOAA, Princeton, NJ, 1992.

8. P. Wang, D. S. Katz, Y. Chao, **Optimization of a parallel ocean general circulation model**, *in the proceedings of the Super Computing 97*, San Jose, California, November, 1997.

9.Yi Chao, P. Peggy Li, Ping Wang, Daniel S. Katz, and Benny N. Cheng, **Ocean modeling and visualization on a massively parallel computer** , *in Parallel Computing for Industrial and Scientific Applications, Morgan Kaufman, 1999.*

Author Biography

Ping Wang *is a Computational Scientist in the Jet Propulsion laboratory, California Institute of Technology. Her research interests include large-scale scientific*

computations, parallel computations in fluid dynamics, parallel software design, and numerical methods (finite volume, finite difference, finite element, multigrid) for PDE. She received her Ph.D. in Applied Mathematics from the City University, London, U.K., 1993. She recently received the best paper award(with Daniel S. Kats and Yi Chao)at SC97

Benny N. Cheng *a Scientific Data Analyst in Earth and Space Science Division of JPL. His interests include numerical analysis of remote sensing data, ocean modeling, and computer visualization. He received his Ph.D. in Mathematics from MIT in 1987, and his Ph.D. in Statistics from UCSB in 1994. Prior to joining JPL in 1996, he was a National Research Council Resident Research Associate.*

Yi Chao *is a Research Scientist in the Earth and Space Science Division of JPL. He has a M.A. and Ph.D. degree in Atmospheric and Oceanic Sciences from Princeton University. Before joining JPL in 1993, he was a post-doctoral fellow in UCLA's Department of Atmospheric Sciences. His research interests are in ocean modeling, satellite remote-sensing of the ocean, air-sea interaction and high-performance computing. In 1996, he received the Lew Allen Award for excellence from JPL, and recently he received the best paper award(with Ping Wang and Daniel S. Kats) at SC97*